[redacted]

510.84
Il6r
no.822 UIUCDCS-R-76-822
Cop 2

Math

THE ROLE OF THE MCDU IN THE PARALLEL ELEMENT PROCESSING ENSEMBLE (PEPE)

by

Charles Paul Yonko

August 1976

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

Report No. UIUCDCS-R-76-822

THE ROLE OF THE MCDU IN THE PARALLEL ELEMENT PROCESSING ENSEMBLE (PEPE)*

by

CHARLES PAUL YONKO

August 1976

University of Illinois at Urbana-Champaign
Department of Computer Science
Urbana, Illinois 61801

*This work was supported in part by the Burroughs Corporation, Paoli, PA 19301, and was submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science, August 1976.

Acknowledgment

I would like to express my appreciation to Dr. David J. Kuck for his help in preparing this paper. This work was performed at the Burroughs Corporation Federal and Special Systems Group, Great Valley Laboratories supported by the Ballistic Missile Defense Advanced Technology Center under contract DAHC60-73-C-0060.

Table of Contents

Processing Ensemble (PEPE) consists of a Control Console
rocessing Elements distributed among eight Bay cabinets.
ree independent computational sub-units and a common mem-
e is organized as three essentially independent control
vailable for inter-control unit communication. The Main-
gnostic Unit (MCDU) is the part of the common logic which
tween the Control Console and (a) the Maintenance Panel,
d Maintenance (T & M) Computer via the PEPE Interface
) the Host or T & M computers via one of the normal PEPE
). These interfaces are established so that the MCDU can
tial system debug and integration, system hardware main-
tenance, failure diagnosis, firmware initialization, and real-time status observation.

The MCDU performs sequential and parallel (simultaneous) functions on PEPE's three parallel associative processors using a combination of shared and dedicated hardware. The MCDU command set has been enhanced through software to perform composite operations beyond the immediate capability of the MCDU hardware. These functions play a major role in the special purpose software system developed to support system debug, maintenance, and diagnostic tasks.

I. Introduction

The Parallel Element Processing Ensemble (PEPE) consists of a Control Console and up to 288 parallel Processing Elements distributed among eight Bay cabinets. Each Element contains three independent computational sub-units and a common memory. The Control Console is organized as three essentially independent control units and common logic available for inter-control unit communication. The Maintenance, Control and Diagnostic Unit (MCDU) is the part of the common logic which provides an interface between the Control Console and (a) the Maintenance Panel, (b) the External Test and Maintenance (T & M) Computer via the PEPE Interface Buffer MCDU port, and (c) the Host or T & M computers via one of the normal PEPE Input/Output Units (IOUs). These interfaces are established so that the MCDU can assist in performing initial system debug and integration, system hardware maintenance, failure diagnosis, firmware initialization, and real-time status observation.

The MCDU performs sequential and parallel (simultaneous) functions on PEPE's three parallel associative processors using a combination of shared and dedicated hardware. The MCDU command set has been enhanced through software to perform composite operations beyond the immediate capability of the MCDU hardware. These functions play a major role in the special purpose software system developed to support system debug, maintenance, and diagnostic tasks.

II. Description of PEPE System

A. PEPE System and Subsystems

The PEPE machine consists of up to eight Bays of Processing Elements arranged symmetrically around the Control Console cabinet (See figure 1). The current MSI implementation includes only one Element Bay containing eleven parallel Processing Elements, although hardware in the control console has been designed and implemented to accommodate a full eight Bay implementation of 288 Elements.

The Control Console interfaces a Host Computer Subsystem through three full-duplex I/O channels. The Host machine is a CDC 7600 with one of its MUX channels directly connected to one Input/Output Unit (IOU) for each processor in PEPE. This interface is established primarily to support application processing

The Control Console also interfaces a Test and Maintenance (T & M) subsystem with four simulated full duplex I/O channels. One channel is dedicated to the MCDU, while the remaining three are assigned to one of the remaining IOU's in each PEPE processor. The T & M Computer is a Burroughs B1714 mini-computer with a special interface provided to simulate the Host-PEPE interface conditions.

B. Test and Maintenance Subsystem

The T & M subsystem is the facility dedicated to debug, integrate, and maintain the PEPE system hardware. The PEPE interface to the T & M computer is via a specialized "black box" called the PEPE Interface Buffer (figure 2). The Buffer is designed to simulate the Host Computer electrically and in function with highspeed I/O. This asynchronous interface is possible due to the 16-bit X 1024-word memory which is loaded from the T & M computer over a simulated disk interface, then transmitted in a burst of 12-bit (16-bit MCDU) parallel words to PEPE. The T & M computer is a standard B1714 microprogrammable computer with a card reader, high speed line printer, two dual disk drives and Supervisory Printer Output (SPO) typewriter. A CRT display terminal has been added to directly support on-line system testing. The T & M computer also supports off-line test and debug using the PEPE Card and Element Tester. The C & E Tester is connected to the B1714 through a second simulated disk interface located in the PEPE Interface Buffer Console, but without the data queueing. The B1700 I/O handler and disk cartridge controller underwent minor modifications to support both of the special interfaces. Under the B1700 Master Control Program (MCPII) mark V.0.4, multiprogramming to support concurrent off-line C & E testing and on-line PEPE system testing is possible.

C. PEPE Functional Description

1. PEPE consists of three essentially independent processors with common logic available for inter-control unit communication. Figure 3 shows the primary inter-unit interfaces. The three global units or processors are called the Arithmetic Control Unit (ACU), the Associative Output Control Unit (AOCU), and the Correlation Control Unit (CCU). The general function assigned to the CCU is that of supplying new data from an external source to the data set distributed in the element memories. The ACU is available for performing arithmetic operations on the data base while retaining a supervisory position as a result of its greater sequential logic resources. The AOCU is, in general, dedicated to providing processed data to an external receiving device.

With certain exceptions the hardware provided each unit is identical and functions in a similar manner.

2. Each global unit is provided with two duplicate Input/Output Units designated IOUØ (Host) and IOU3 (T & M), with expansion capability reserved for two additional IOUs. Each IOU consists of an independent read and write channel to provide full-duplex, 12-bit parallel I/O. All four channels share direct-memory access to both program and data memories. This access can be initiated either by sequential instruction execution or by an IOU command from an external device to block transfer data. Five data conversion formats are available for transforming CDC 7600 one's complement 60-bit words to one of the PEPE two's complement formats. The IOUs also have clear, stop, start and interrupt capabilities. Each IOU read channel can send data and commands to the MCDU when its IOU mode of operation is enabled. Under the control of the MCDU, IOUØ and IOU3 can be configured into one of several "Control Functions" to support both hardware and software debug and failure diagnosis. Each global unit also contains an IOU-system interface unit which provides inter-IOU priority resolution for memory access and IOU interrupts.

3. Each global unit is provided with a program memory, denoted S*PGRM, and a data memory, denoted S*DATA, where "*" is replaced by "A"-ACU, "C"-CCU, "O"-AOCU in figure 3. The program memory is designed to contain sequential and parallel instructions. Operands may be routed and data stored at the expense of an execution time penalty due to instruction fetch ahead which normally overlaps operand fetch. Data memory is dedicated to data base support. All memories have a 100 ns access time except SAPGRM which is 200 ns due to its much larger size. All memories are capable of expansion to 4K 32-bit words except SAPGRM which can be expanded to 64K. Each global memory address bus, denoted S*PAIN and S*DAIN for program and data memory respectively, is routed directly to the MCDU for breakpoint operations and real-time display.

4. The sequential control logic (SCL) executes programs from program memory with sequential instructions executed locally while routing parallel instructions and operands to the Parallel Instruction Control Unit (PICU). The ACU is designated more powerful than the other two control units with its capability to initiate parallel floating point arithmetic, and perform interrupt handling operations scheduled by the interrupt control hardware located in the Inter-Communication Logic (ICL). Sequential execution is based on hardware enables from the microprogram memories which are initialized as an MCDU task.

5. The Parallel Instruction Queue (PIQ) was placed between the ACU-SCL and the ACU-PICU because of the extended duration of floating point execution in the parallel Arithmetic Units. It can queue up to 16 parallel instructions with operands in the event that the PICU falls so far behind the SCL that the SCL must wait to complete another parallel instruction routing.

6. The Parallel Instruction Control Unit (PICU) accepts instructions and operands from the SCL and generates the execution enables from its micro-program memory which are distributed to all of its parallel counterparts. Since each parallel processing element contains three execution units but only a single memory, instructions which require parallel memory usage are resolved at the PICU level via its interface with the Element Memory Control (EMC). Similarly, operations which retrieve data from a processing element over the shared Output Data Bus must be resolved at the SCL and PICU level over the Output Data Control (ODC) interface.

7. The actual interface between the Control Console and the several element bays is done by the Signal Distribution System which consists of the Central Signal Distributor (CSD) and the Bay Signal Distributor (BSD). In addition to signal fan-out, logic is included to support associative processing and to provide status information for other functions in the Control Console. The CSD is physically in the Control Console.

8. Each Bay consists of a Bay Signal Distributor (BSD) and up to 36 identical parallel Processing Elements. Each Element contains a memory and three processing units, corresponding to the three global control units, which can be independently enabled to take part in its current instruction stream. Output from the element is via a data bus shared by the parallel Arithmetic Unit (AU) and Associative Output Unit (AOU), while the Correlation Unit (CU) cannot output data to the Control Console.

9. The Inter-Communication Logic (ICL) is the largest portion of the common logic in the Control Console. It provides the mechanism for all inter-unit interrupts, Real Time Clock and Interval Timer functions, system data gathering functions and error control.

10. The Maintenance Panel provides real-time visual system status and control (figure 4). Its major interface is the MCDU which executes commands and processes data entered into its switches. It also provides control to, and status of power and system clock.

11. The MCDU is the part of the common Control Console logic which provides system control via one of its modes of operation selected by a maintenance panel switch. It is provided with a set of commands which can be executed from either the Maintenance Panel, directly from the T & M computer via the Interface Buffer MCDU Port, or any one of the six IOU's.

III. The MCDU

A. System Requirement

The primary system need for the MCDU is for microprogram memory (MPM) initialization. All six MPMs have been implemented in using 1024 X 1-bit random access memory (RAM) chips for the required execution enables (80 bits maximum), and must be reinitialized at every power-up. The remainder of the MCDU functions support both more complete system initialization, system maintenance and diagnostic operations, and can be used for real-time status observation. These functions are accomplished by combining a set of MCDU commands and data in a manner which is essentially identical if done manually from the Panel or automatically from a remote device.

B. The Command Set

An MCDU command consists of four fields which partition the 16-bit word. The first two fields, COMMAND and VARiant1, specify the operation while

15 12	11 9	8 5	4 Ø
COMMAND	VAR1	VAR2	ADDRESS

the second two, VARiant2 and ADDRESS, are used to specify a destination within PEPE. The format was selected to simplify comprehension of variations in hardware functions while maintaining destination codes compatible with codes established within the PEPE architecture. Octal codes are used for consistency with existing system documentation.

1. Command ØØ$_8$: NO-OP

This command is provided to simplify software structures consisting of a generalized sequence of MCDU commands, some of which may be optionally specified without affecting the execution of the remainder of the sequence. As a stand alone command, it allows exercising of bits 11-Ø of the input bussing and command register, since these fields are unused, and results in no additional modification of the MCDU state.

2. Command Ø1$_8$: STATIC TEST

"Static" because the referenced PEPE control units cannot be running, this command has four variants.

a. LOAD and DUMP can access all SCL registers. Any PIQ word and any pre-selected 16-bit group of signals from the CSD can be DUMPed. LOAD and DUMP are performed sequentially by the MCDU assuming control over various shared and dedicated portions of the Control Console.

To DUMP an SCL register, the MCDU provides an address to the associated microprogram memory word and causes the SCL to process it to the point of presenting the data on its normal system bidirectional data bus to the ICL (bus B in figure 6). The MCDU then controls the bus and portions of the ICL data switching network to route the data to the bidirectional ICL - MCDU data bus (bus C in figure 6) and into the MCDU Output Register.

The MCDU LOAD's SCL registers in a similar manner by enabling the Input Register to the ICL via bus "C", through the ICL data switching network to the normal bidirectional bus of the selected SCL. A separate microprogram memory word enables data into each register under MCDU control.

LOADing and DUMPing any SCL register does not interfere with the current state of that SCL nor does it affect any other SCL's which may be running. The MCDU obtains its control via the normal ICL interrupt control mechanism with the MCDU holding the highest interrupt priority and cannot be inhibited.

PIQ words are DUMPed in 32-bit parallel halves with the most significant half zero filled. The mechanism is similar to dumping an SCL register except that the selected PIQ destination is presented to the uni-directional PIQ-ICL data bus before routing to the MCDU.

The CSD contains logic to select among the various parallel execution enables and data in 16-bit groups under direct MCDU control. This allows access to PICU outputs, Element Memory address, Element status and activity, and data returned via the Output Data Bus at the Control Console - Element Bay interface.

b. The SINGLE STEP variant enables one complete, simultaneous execution cycle, including operand routing for all specified global units. If the Instruction Buffer of a referenced SCL has been loaded from the MCDU, that instruction will be executed. Otherwise, the Instruction Buffer will be loaded from program memory in the normal manner allowing programs to be STEPped in parallel, independent of the sequential/parallel instruction mix and duration in each global unit. Interleaved MCDU data sampling is permitted since the "STATIC" state is maintained by the MCDU detecting the completion of the longest duration instruction in any of the three global units before completing the STEP.

c. The SINGLE MICROSTEP variant allows the MCDU to simultaneously execute one microstep in all active parallel processing units for any combination of the specified global units. The MCDU places all referenced PICU's in a single microstep mode and initiates a SINGLE STEP operation in the respective SCL's to route parallel instruction information and operands to the PICU. Since all SCL instructions are currently one microstep, embedded sequential instructions will execute to completion while parallel instructions which cannot be accepted by a busy PICU will exempt the affected SCL from participating in the current single microstep operation.

The MCDU must detect this latter condition so that completion of the micro-step operation in the MCDU is tied to the PICU execution duration rather than completion of the SCL step operation which the MCDU must attempt to complete as an independent operation to maintain the static state. Inter-leaved CSD data sampling is allowed.

3. Command $\emptyset 2_8$: ENABLE MICROPROGRAM MEMORY

This command allows four varied operations on any one of PEPE's six MPMs. Each MPM is considered as 1024 words X 80 bits although not all enables are implemented. Each word is treated as five 16-bit segments. An address/segment counter is provided in each MPM which can be controlled from the MCDU. Two dedicated bidirectional "party-line" data busses are used for MPM operations which can be performed only when PEPE is stopped. Four control signals are "daisy-chained" in an analogous fashion with separate select signals provided to enable each MPM to respond individually.

a. RESET ADDRESS/SEGMENT COUNTERS initializes the distributed MPM control to the most significant segment of word zero in only the one of six MPMs selected. It is usual, but not necessary, to begin MPM operations with this variant.

b. RUN UNTIL ADDRESS COMPARE causes the address/segment counter of the selected MPM to increment at the system clock rate. The address register is gated via one of the bidirectional MPM busses to the MCDU where the comparator is used to detect the address contained in the least significant half of the MCDU Input Register. When found, the segment counter is initialized to the most significant segment at that location by trailing edge detecting the "RUN" control. This operation can begin and end at any MPM address with wrap-around to address zero occurring normally after the least significant segment of address 1777_8. The typical application is when a single MPM word must be modified.

c. LOAD and DUMP are performed to the currently addressed segment, with each access automatically incrementing the segment pointer. Data is loaded from the least significant half of the Input Register and DUMPed to Output Register(15 - Ø). The mode signal is used to control the data bus enables as well as gating a one clock width pulse from the MCDU to be used as a memory write enable. These controls affect only the MPM whose select signal is true.

4. Command $Ø3_8$: STOP ON GLOBAL MEMORY ADDRESS

This command provides breakpoint capability to stop all running global units or only the unit whose program or data memory address bus is being monitored. The MCDU selects one of the six global memory address busses, which are driven directly from the memory, into the comparator for detection of the address contained in the least significant half of the Input Register. On address detection, a stop request is transmitted to the unit whose memory address is being monitored, or to all units as a VARiantl option. The MCDU will complete execution when the last unit involved has stopped after completing execution of the instruction pending when the stop request occurred.

5. Command $Ø4_8$: STOP ON SUPERVISORY LOAD

This command will monitor control unit activity for execution of a Supervisory Load instruction by any of the specified global units. On detection, only those units monitored or all units will be stopped as a VARiantl option. A Supervisory Load instruction is one of the major system vehicles for inter-unit communication and access to common resources located in the ICL. The MCDU will not complete execution of command Ø4 until all units involved have been stopped.

6. Command $Ø5_8$: Unconditional STOP/START

This command is performed by MCDU control of the SCL start and stop signals originating in the ICL.

a. The STOP variant issues simultaneous stop requests to all specified units. Any pending system start requests will be inhibited and the referenced SCL's will stop at the end of their currently executing instruction. If more than one unit is requested to stop, the MCDU will not complete execution until the longest duration instruction has completed in the referenced SCL. If a referenced unit is already stopped this variant acts as no-op for that unit.

b. The START variant issues simultaneous start requests to all specified units. Any pending stop requests are inhibited or removed so that the referenced SCL can begin execution on the "next" system clock. Execution will begin by loading the Instruction Buffer from the current Program Memory location specified by the Program Counter unless the Instruction Buffer was loaded from the MCDU. In this case, the Instruction Buffer contents will be executed before being loaded from the (possibly new) Program Counter address. The MCDU will complete execution of this command when all requested units have responded to the start.

c. A DELAY START variant was added to the command set to account for the delay between starting PEPE execution and preparing a breakpoint operation. This variant allows the MCDU to accept one additional command before starting the specified control units.

7. Command $Ø6_8$: SET IOU CONTROL FUNCTION

The MCDU contains three four-bit registers for maintaining a binary code transmitted to the IOUs in each of the three control units respectively. Each IOU channel decodes the pattern to determine its configuration and degree of participation in the selected "Control Function". The MCDU sets the code for all units specified and returns it to the "Normal" code when a Master Clear of the global unit is detected. The control functions allow Host and T & M computer inter-communication, recording of PEPE-HOST and PEPE-T & M communication by the "odd" computer, interface testing without involving conversion and other normal IOU logic, and the ability of each computer to use the IOU logic assigned to the other computer as both a diagnostic aid and a backup interface which can be programmatically configured via the MCDU.

8. Command $1\emptyset_8$: MASTER CLEAR issues a clear signal to the ICL logic which performs a system clear cycle for only those units specified. A clear signal for each unit is returned to the MCDU to initialize the IOU control function. If the MCDU detects that all units are being cleared simultaneously, the MCDU begins a local clear cycle.

9. Command 11_8: MCDU TURNAROUND has four variants to assist with interface and local MCDU testing and can be used to dynamically sample system status.

a. CLEAR MCDU initiates an MCDU clear cycle to return all flip-flops to their inactive state and loads all registers with zeros. The remainder of PEPE is unaffected including the IOU Control Function registers.

b. INPUT REGISTER TRANSFER causes the Output Register to be loaded in a single 32-bit parallel operation from the Input Register.

c. COMMAND REGISTER TRANSFER loads the least significant half of the Output Register from the 16-bit Command Register and zeros the most significant 16 bits.

d. DYNAMIC DUMP causes the MCDU Output Register to be loaded with the current contents of the selected destination although PEPE may be in real-time operation. The destinations include the System Mode Register (SAXMDE in figure 6) which maintains the status of the control units, any global memory address bus or any 16-bit group from the CSD.

10. Command 12_8: STOP ON ELEMENT MEMORY ADDRESS will stop all units or only those units which are monitored for accessing parallel Element Memory at the address contained in the MCDU Input Register. The MCDU provides a four bit address to the CSD to select the 16-bit group of signals containing the Element Memory Address bus. Since this group contains other signals, the MCDU activates a mask in its comparator enabling only the address bits to generate the stop condition. The MCDU then initiates the stop requests if it detects that the Element Memory Control has assigned priority to one of the units being monitored. If the ACU is specified, the PIQ will be prevented from continuing until empty.

11. Command 13_8: STOP ON ERROR causes the MCDU to activate a signal to the ICL which inhibits errors from the normal system handling mechanism. When an error occurs in one of the units specified in the MCDU command, all units will be stopped, including the PIQ which is prevented from continuing until empty.

12. Command 16_8: REPEAT will continuously execute the last instruction entered into the Instruction Buffers of all specified control units. Execution includes routing of data to form the operand and the actual function of the instruction except that the Instruction Buffer cannot be loaded (one of the operand routings allows instruction self modification). The REPEAT state can be exited by a Master Clear of all units although an MCDU clear will stop all affected units as part of clearing the repeat flip-flops.

13. Command 17_8: DATA COUNT is available primarily for communication to an external computer. The control signal discipline for the MCDU is similar to that of the IOU (see figure 8). Information present on the input channel when the MCDU Word Counter is empty is treated as a command and loaded into the Command Register. If the word count is nonzero, the specified count is the number of 16-bit words of data to be accepted. When the Output Register is loaded, an output flag is set so that the next Data Count command will load the Word Counter with the count of the number of 16-bit words to be placed on the output channel.

The first variant causes Command Register bits (5-∅) to be transferred to the Word Counter with zero fill. Another variant causes the MCDU to accept one additional word on its input channel to be used as the word count. When the MCDU is controlled by the Maintenance Panel, this command is unnecessary since data is directly entered into the Input Register, and the Output Register is the final destination for data from PEPE.

Data on the input channel is toggled into halves of the Input Register to form a 32-bit PEPE word, except when an MPM load operation is enabled. Command 17_8 must follow an MPM Load Enable so that the block of data may be buffered in the least significant half of the Input Register 16 bits at a time. With output channel operations, the MCDU detects the size of its interface with PEPE so that 32-bit words are toggled out of the Output Register most significant half first, and 16-bit words will always be from the least significant half of the register.

A VARiantl bit allows an output channel Record pulse to be issued when the word count becomes zero. This signals the Buffer that transmission is complete. During a Conversational interaction involving a number of unique output events from the MCDU, only the last DATA COUNT Command can have this Record option set without terminating prematurely.

14. Commands Ø7, 14 and 15 are currently undefined.

C. MCDU Architecture

The MCDU contains a 16-bit Command Register (figure 5) which is decoded using conventional "hard-wired" logic to provide the local and global controls required by the command set. It is loaded in 16-bit parallel from the output of Control Point One (CP1). The Input Register is a 32-bit data register which is loaded in halves or in 32-bit parallel from the Maintenance Panel switches as selected by Control Point Two. The 16-bit Word Count Register is loaded from the Command Register bits 5-Ø with zero fill or a 16-bit count from CPØ via CP3. The Word Counter is a synchronous down counter with built in zero detection. All of these registers have drivers for LED indicators on the Maintenance Panel. The Output Register is loaded in 32-bit parallel with zero fill if necessary from the output of Control Point 4 which selects among the six global memory address busses, the Command Register, Input Register, a 16-bit bus from the CSD, the 32-bit bi-directional data bus to the remainder of the Control Console via the ICL, and a duplicated 16-bit bi-directional bus used for microprogram memory operations. A 16-bit comparator with a fixed mask is used to detect the contents of the Input Register (15 - Ø) occurring on one of the input busses to Control Point 4. Control Point 5 selects halves of the Output Register onto the bus which returns to the Interface Buffer. Control Point 6 selects between the MCDU Output Register and one of the inputs to Control Point 4, as determined by the "DISPLAY" rotary switch on the Maintenance Panel (figure 4). Control Point Ø selects either the bus from the Interface Buffer or the least significant 16 bits of the bus to the ICL as a function of the MCDU mode enabled by another Panel switch.

D. Special Cases of Operation

1. Interface Buffer

The Interface Buffer was designed to simulate a CDC 7600 MUX channel so that the IOU's could be of a common design and could be tested in a simulated full speed asynchronous environment (approximately one 12-bit parallel word every 500 nanoseconds at zero cable length). The IOU can acknowledge a response from the Buffer without retiming, however all MCDU I/O retimes the Buffer control signals at a loss of at worst two (10 MHZ) clocks. The primary mode of Buffer operation is a block transfer Read or Write in which the Buffer is locked onto a single read or single write channel among the 8 possibilities, according to a predefined priority scheme with the MCDU holding the highest priority. The Buffer also has a Conversational mode of operation (figure 8) available only to the MCDU. The Buffer memory is dynamically repartitioned into a write portion and a read portion having a 10 to 1 size ratio. Commands and data are output from the write portion to the MCDU and may result in interleaved input to the read portion at speeds comparable to the block transfer modes. Thus a high speed predefined sequence of parallel operations and interrogations may be performed using the MCDU without the interference of the T & M Computer with its much slower interface and I/O overhead otherwise required for one-at-a-time PEPE data acquisition.

2. Maintenance Panel

The MCDU can be controlled from the Maintenance Panel which reflects much of the MCDU hardware. Data is entered directly into the Input Register from the Panel data switches (figure 4) by activating the associated enter pushbutton. The Command Register is loaded in the same manner except that an MCDU execution cycle is automatically initiated. If the MCDU were "hung" on a previous command, a new execution cycle can be forced by depressing the enter command pushbutton. Commands which load data use the current contents of the Input Register while dump type commands which load the Output Register must have the Panel DISPLAY rotary switch in the MCDU position to observe the results in the DISPLAY LEDs. The STATUS LEDs are driven from the MCDU which generates some of the displayed information while the remainder is generated in real-time from other areas of the Control Console. The Parallel Activity Count display is driven from the MCDU as an independent real-time function which converts the separate binary counts to BCD and provides continuous refresh of the nine Panaplex numerals.

When executing from the Panel, the Data Count command 17_8 is unnecessary. It can be executed to load the Word Counter from bits 5 - Ø of the Command Register and is decremented by depressing ENTER associated with the Input Register as a local test of its operation. The Enable MPM Load and Dump command ($Ø2_8$) variants also execute differently from the Panel. The command must be executed for each 16-bit MPM segment accessed, although the normal MPM address/segment counter operation remains unaffected.

3. Input/Output Unit Enabled

Each IOUØ and IOU3 of every control unit is capable of routing commands and data to the MCDU via the least significant 16 bits of it unidirectional interrupt data bus ("A" in figure 6). The primary requirement is for initializing the microprogram memories from any compatible external device as a backup to the normal operation from the T & M computer. The MCDU does not restrict the execution of any command accepted from this interface, however since output channel controls have not been provided, any attempts to output from the MCDU to the IOU will result in a "hung" condition.

To send a word to the MCDU the IOU must request priority for the shared inter-IOU interrupt data bus. When granted, the data is placed on the bus ("A" in figure 6) by the IOU System Interface Unit which informs the MCDU that data is available. The MCDU controls ICL Control Point 18 to route the data to MCDU Control Point Ø of that bidirectional interface ("C"). At that point, the MCDU acts in the usual manner to distinguish commands and data. When execution of the command is complete, the MCDU will acknowledge the IOU via the normal system interrupt mechanism indicating that another word may be issued.

E. Hardware Partition

The MCDU consists of five PEPE six-layer composite circuit boards which contain printed circuit and a variety of descrete wire conductors. Each board contains 300 dual-in-line package (DIP) sockets with 16 wire-wrappable pins. There are two versions of the composite board, one which allows backplane connection to belted ribbon cable for one connector, the other having four connectors configured for backplane wiring with subminiature coax wire. Each board has 232 backplane pins (264 with cable connector) available for signals. The MCDU uses over 900 DIPs including MECL 10,000 logic devices, delay modules and a variety of resistor modules. Backplane pin usage averages 225 signals per board.

The MCDU is partitioned into two identical data boards and three control boards. The data boards (figure 7) each contain two quarters of a 32-bit word. This arrangement simplifies toggling halves of a 32-bit word to the output channel while duplicated hardware is a definite manufacturing (cost) advantage. The bi-directional MPM data bus is duplicated to account for MPM distribution between two separate power supplies in the Control Console. Each 8-bit slice for the comparator generates an "equal" condition which is "AND"ed on control board MU3 to generate address trap conditions.

Control board MU3 contains the Command Register, Word Count Register and the majority of the command control logic. SCL start and stop control signals eminate from MU3.

The Method of controlling the start/stop controls partitions the STATIC TEST and conditional stop commands into three groups. Static Test Load and Dump use a special condition of start and stop simultaneously true in executing from MPM. The Step operations initiate the start requests on system clock but must issue a stop request asynchronously on detecting that the execution has begun, and must prevent the occurrence of the special Load/Dump conditions. Unconditional Start also begins on a system clock but does not require immediate removal of the enable when execution begins. Both of these start conditions remove pending system stop requests to prevent a Static Load/Dump occurrence. Unconditional Stop is implemented as the immediate detection of a stop condition. The occurrence of one of the stop conditions (Error, Global Memory Address, unconditional, Element Memory Address, Supervisory Load instruction) immediately triggers a stop request for each affected unit as specified in the command variants. The three groups of stop requests and start requests are simply "OR"ed for their respective units and sent to the SCLs via the ICL which cannot block the MCDU signals.

MU4 contains logic for Set IOU Control Function, Master Clear and portions of the MPM enable commands. Output from the Panel Clear pushbuttons antibounce circuitry is shared with the Master Clear command logic. All drivers and receivers for the belted cable to the Interface Buffer as well as execution timing pulse generation logic is located on MU4.

MU5 has a belted cable connection to the CSD with receivers for the three busses containing the count of active parallel processing units. Logic to perform the binary to BCD conversion and to drive the panaplex display is provided. The data bus from the CSD which contains selected 16-bit groups of data also shares this belted cable and is routed to the data boards via MU5. Local multiplexing of controls for Control Point 4 is performed on MU5. The Panel DISPLAY rotary switch determines a default address for CP4 when no higher priority useage is involved. In the "MCDU" position, the Output Register is addressed. In the "S--AIN" position, the Panel Command Register data switches (8-5) are enabled to generate the address of the desired global memory address bus. In the "CSD" position, the 16-bit CSD data bus address is selected and the Panel Command Register data switches (4-Ø) are enabled to the CSD. The "SAXMDE" position selects the System Mode Register onto the MCDU-ICL data bus via ICL Control Point 18. A second level of CP4 address multiplexing is provided to preempt the display address when the MCDU IOU mode of input is enabled, but only when data is about to be accepted from the IOU. This leaves the real-time DISPLAY intact. A third level of address selection for CP4 is tied to command execution which assumes ultimate data control only when required, again leaving the real-time DISPLAY visibly unaltered. Thus it is possible to load an SCL register from the IOU via the MCDU while an uninterrupted display of the System Mode Register is observed, all operations time sharing effectively the same hardware. Drivers for the Panel STATUS LED's are also located on MU5.

Each PEPE board is provided with a test point module with 10 locations available for switches, LEDs or coaxial connectors. One coax connector must be assigned to clock and is adjusted to account for signal skew. The MCDU contains two switches, one enables a pulse generator driven from system clock, the second enables the MCDU to cycle based on one of those pulses. One LED indicates that an MCDU execution cycle has been initiated and is extinguished as a result of completing the pending command. Another indicates that an output operation was initiated and requires an acknowledge. A third LED indicates that a stop condition has been detected and requires a PEPE unit to be stopped. Additional LEDs and coax jacks are used to indicate local multiplexor addressing and bussing control.

IV. Application

A. MSS Support Software

In lieu of writing debug, maintenance and diagnostic tests using a commercial general purpose programming language, a special purpose software package was developed to reflect the PEPE - T & M environment and simplify the test programming task. The Maintenance System Software (MSS) structure was designed to provide modular growth to the set of system test support software to match the availability of system hardware. Since the MCDU was the first available system hardware and remained the primary I/O and control device for "boot-strapping" the hardware integration, the initial debug support software, RAID, is very much MCDU oriented. Subsequent MSS modules continued to rely on techniques developed for using the MCDU.

B. RAID Software Debug Tool

The Real-time Aid for Interactive Debugging (RAID) software became available when the MCDU completed fabrication as the first PEPE system hardware. A framework of "Control Verbs" to direct T & M processing and "Process Verbs" to perform functions on PEPE via the MCDU was defined. The initial implementation provided translation of octal input into binary and immediate execution using the LIT Process Verb (Table 1). Soon afterward, additional mnemonic Process Verbs were made available as more convenient forms of the corresponding MCDU commands. PEPE registers directly available to the MCDU were given mnemonics compatible with their hardware names.

The method of loading an SCL register from the T & M computer consists of the following sequence of MCDU commands and data in 16-bit parallel:

i DATA COUNT COMMAND (to load Input Register)
ii MOST SIGNIFICANT HALF DATA
iii LEAST SIGNIFICANT HALF DATA
iv STATIC TEST LOAD TO SPECIFIED SCL REGISTER

On receiving an input message to perform "LOAD SCAREG 12345", RAID immediately converts the last field to binary and enters this data into items ii and iii. The "C" and "AREG" result in the CCU and A-register codes entered into the Variant2 and Address fields of item iv respectively. The entire sequence is preceded with an Interface Buffer command specifying Conversational Mode to write four words via its MCDU port. RAID initiates a B1700 I/O to test the interface and to perform the write to the Buffer. RAID will then continue to interrogate its interface to detect completion of the write to PEPE based on the I/O descriptor provided by Buffer status signals sent via its disk cartridge controller. The number of interrogations performed before a time-out is signalled can be specified by the user. On detection of a successful I/O, RAID returns to the user for the next input message. The DUMP Process Verb executes in an analogous manner except that the resulting PEPE output sets a bit in the I/O descriptor which indicates that the Buffer memory must be read. RAID then initiates a read I/O to address the "read-back" portion of the returned memory contents, converts and displays the data to the user.

As more PEPE hardware became available, the first composite Process Verb was implemented by incorporating PEPE instructions. To LOAD or DUMP the contents of a global memory location, the following operations must be performed via the MCDU:

I. LOAD SEQUENTIAL A-REGISTER WITH DATA
II. LOAD INSTRUCTION BUFFER WITH STORE A-REGISTER INSTRUCTION
III. SINGLE STEP EXECUTE

I and II are represented by the preceding generalized sequence to load an SCL register. "LOAD SAPMEM 12345 65432" will have the last field loaded into the sequential A-register, a store A-register instruction to program memory address 12345 loaded into the ACU Instruction Buffer, and executed under MCDU control. The I/O operation is identical to the single register operation except that all three events are performed in one write to the Buffer.

This method can be expanded to access any PEPE destination including parallel registers, for example "DUMP POAREG", although these have not yet been implemented.

RAID was further expanded to include basic file handling operations so that more comprehensive tests could be developed, and to take advantage of the speed available at the interface.

C. SURPASS Assembler

RAID was further enhanced by the development of an assembler for PEPE instructions, acronym SURPASS, which creates non-relocatable object modules for the various PEPE global memories. These modules can be loaded into PEPE using the RAID "XFER" Process Verb as a unique event or as part of a user created "Active File" (figure 9).

Test "Segment" ARINTC from file PEPE.CODE/RQITRI created by SURPASS (figure 10) is loaded into consecutive ACU program and data memory locations via the MCDU while CRQITRI is loaded via CCU-IOU3 when RAID executes the active file. After being started by the MCDU, the PEPE resident program will return a result descriptor which is tested by RAID, and which prints the Buffer contents on failure.

The MCDU Loader provides speed comparable to that for the IØU because of a small PEPE resident loader in association with the T & M which reduces the number of MCDU operations below the number normally required to LOAD a memory location. The PEPE-resident portion consists of the following operations:

i STORE A-REGISTER IN STARTING ADDRESS OFFSET BY INDEX REGISTER
ii INCREMENT INDEX REGISTER
iii STOP
iv JUMP TO STORE INSTRUCTION

After preserving the PEPE state and initializing the loader parameters, the only MCDU operations required are:

I LOAD A-REGISTER
II START EXECUTION

Since the SCL program counter points to the "next" instruction, every MCDU start will begin execution with the Jump in item iv above, eliminating the LOAD INSTRUCTION BUFFER operation from the usual MCDU sequence.

D. MODEST Translator and DOC Interpreter

The Diagnostic and Operability Control Program (DOC) and the Modular Operability/Diagnostic Execution Statement Translator (MODEST) comprise the remaining portions of the MSS structure. MODEST is a high level problem oriented language that provides more user flexibility in writing test programs. MODEST creates an "S-Language" disk file containing the translated test execution statements. The S-Language is then "executed" by a pseudo-machine, represented by DOC, whose working memory represents the Interface Buffer. MODEST features include integer arithmetic and logical functions, conditional branching, subroutine creation and linkage, output to the system display, line printer or B1700 Master Control Program, and test debug features. Using the subroutine features, a procedure can be created at the MCDU command level and sent to PEPE by storing into the pseudo machine memory. The results can then be read from the Buffer to allow pseudo-memory fetches for further test manipulation using DOC's 32 pseudo-registers. MODEST also provides for loading and execution of PEPE resident programs, created by the SURPASS assembler, the results of which can be automatically analyzed by the users T & M resident MODEST program. DOC is implemented within the RAID program to take advantage of the established PEPE/T & M routines.

V. Summary

The MCDU was the first operational PEPE system hardware and continues to be a primary I/O and control device throughout system integration and on-going system maintenance and diagnostic work. MCDU commands, which involve a combination of dedicated and shared hardware throughout the PEPE Control Console, perform both sequential and parallel functions on PEPE's three parallel associative processors. A special purpose software system has been developed to simplify the hardware system integration and maintenance task by capitalizing on the control centralized in the MCDU. Inherent flexibility to expand the basic functions performed through the software by further combining PEPE instructions under MCDU control adds a different perspective to this entry into parallel processing.

References

1. A. J. Evensen, and J. L. Troy, "Introduction to the Architecture of a 288-Element PEPE", 1973 Sagamore Computer Conference on Parallel Processing.

2. PEPE System Functional Design Specification, Vol. II, Hardware Specification, System Development Corporation, Revision E, October 1975.

3. C. P. Yonko, MCDU Command Set, Attachment to Burroughs Corporation Memorandum #33500-74-135, 16 July 1974.

4. CONTROL DATA 7600/CYBER 70 Model 76
Computer Systems Hardware Reference Manual, Revision A, 1972.

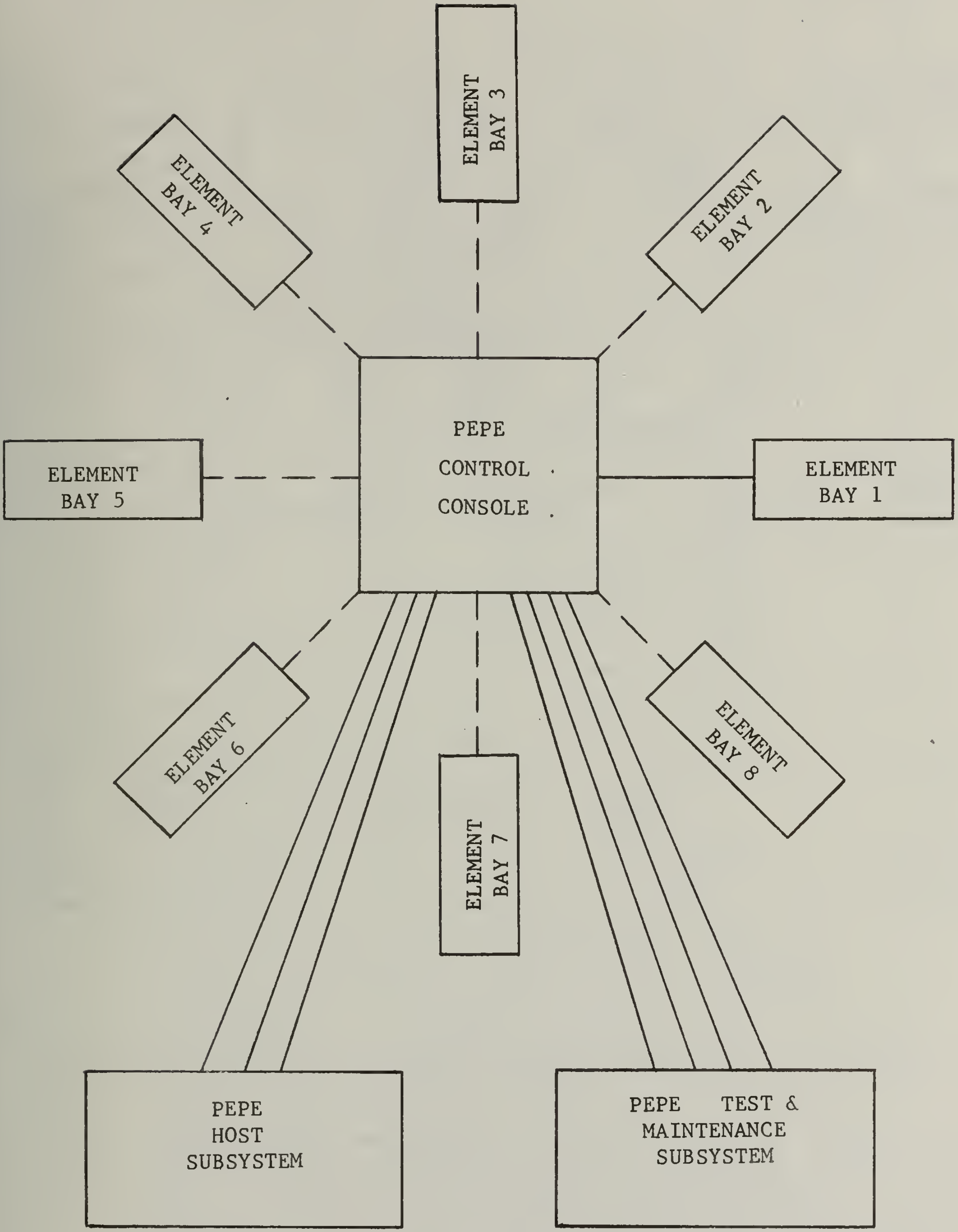

Figure 1. PEPE System and Support Subsystems Configuration

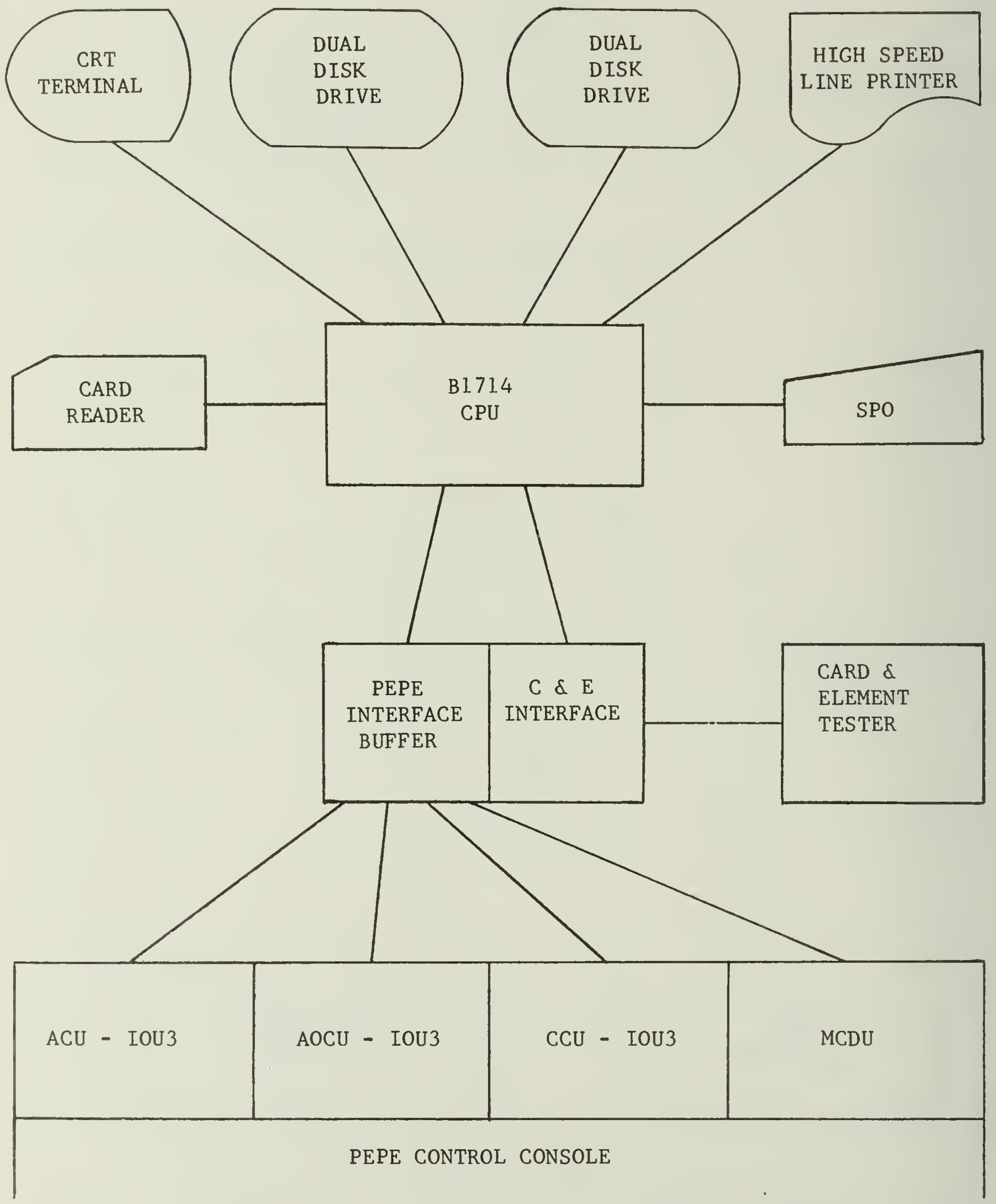

Figure 2. PEPE Test and Maintenance Subsystem

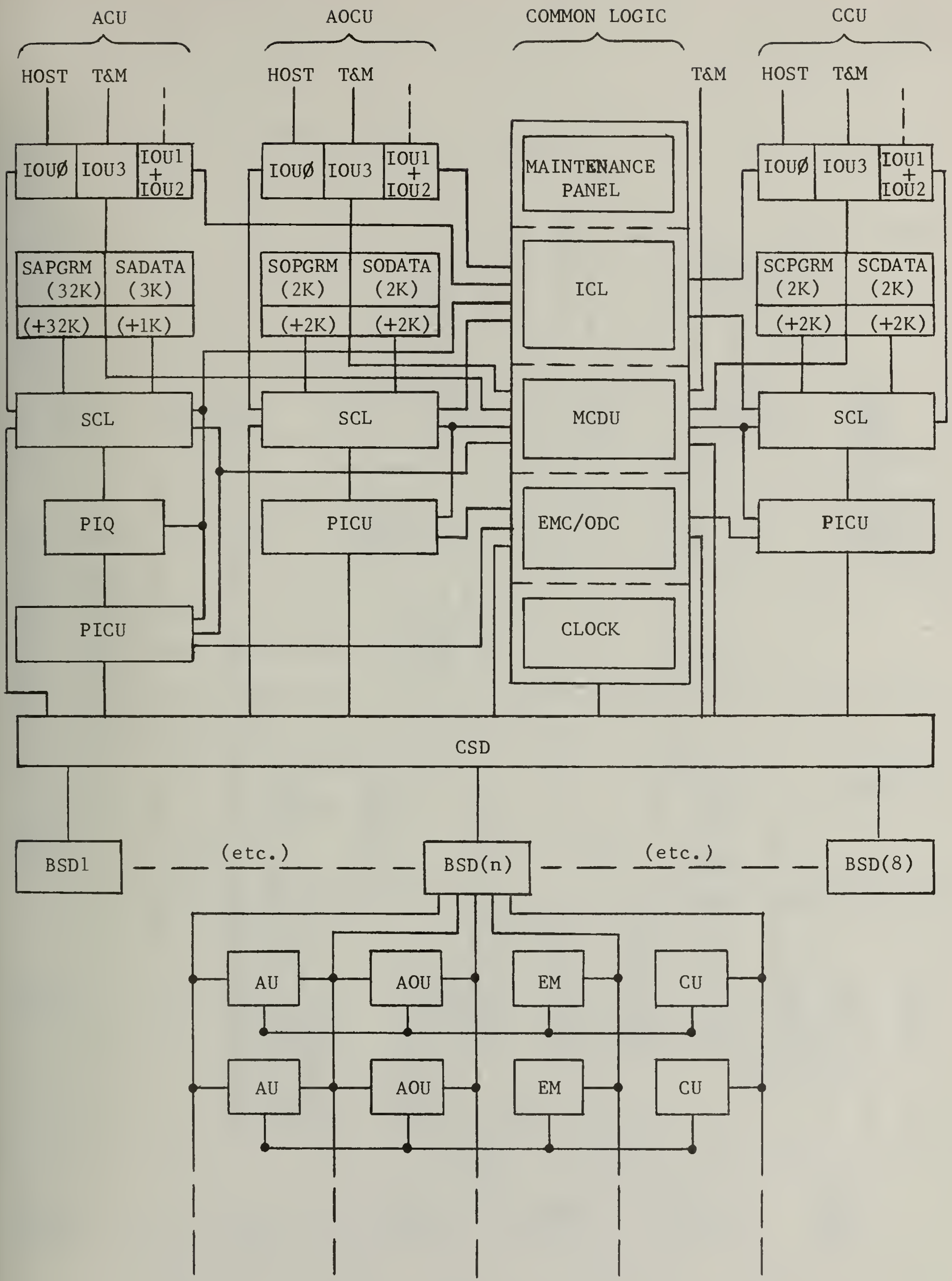

Figure 3. PEPE System Block Diagram

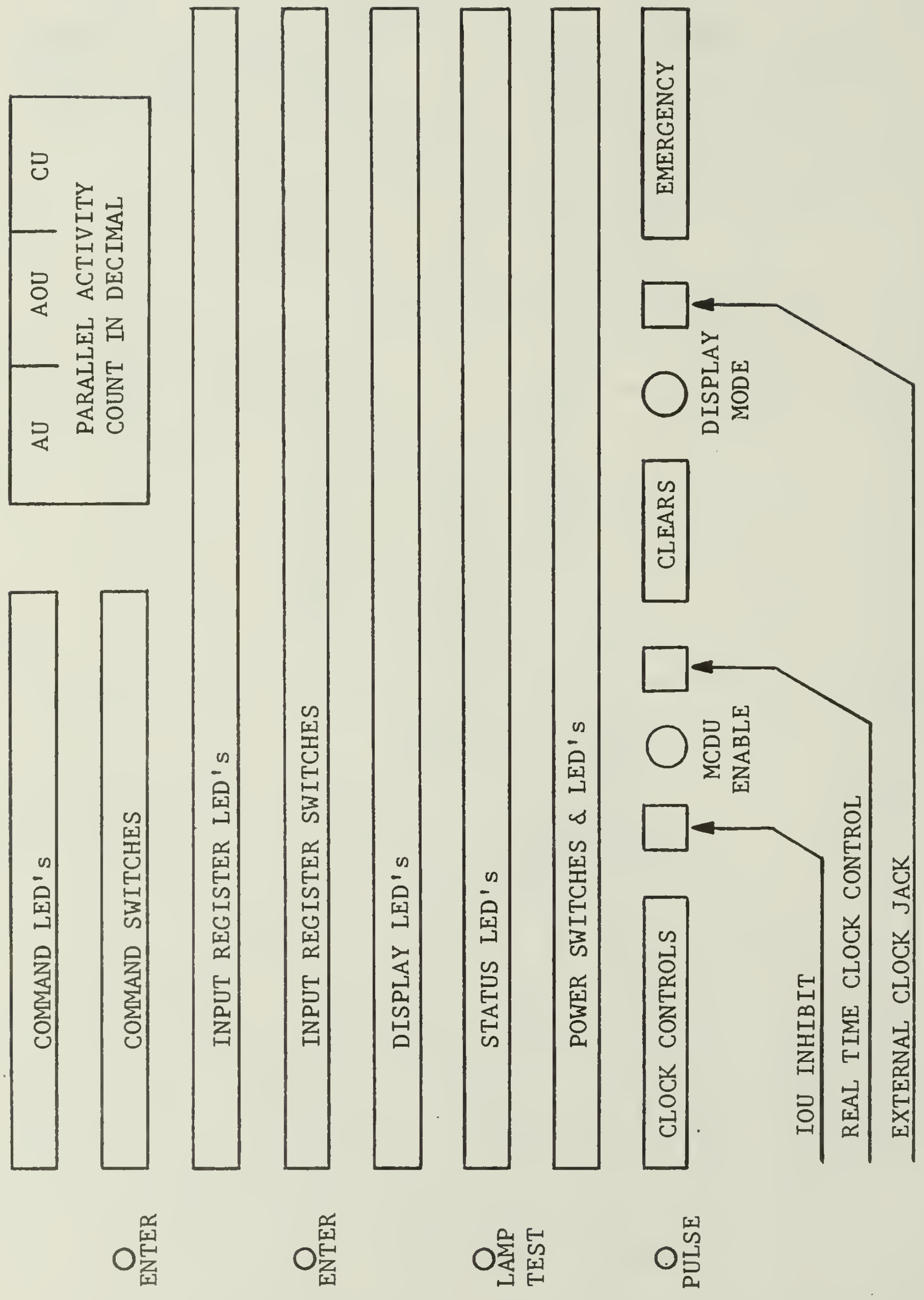

Figure 4. Functions On Maintenance Panel

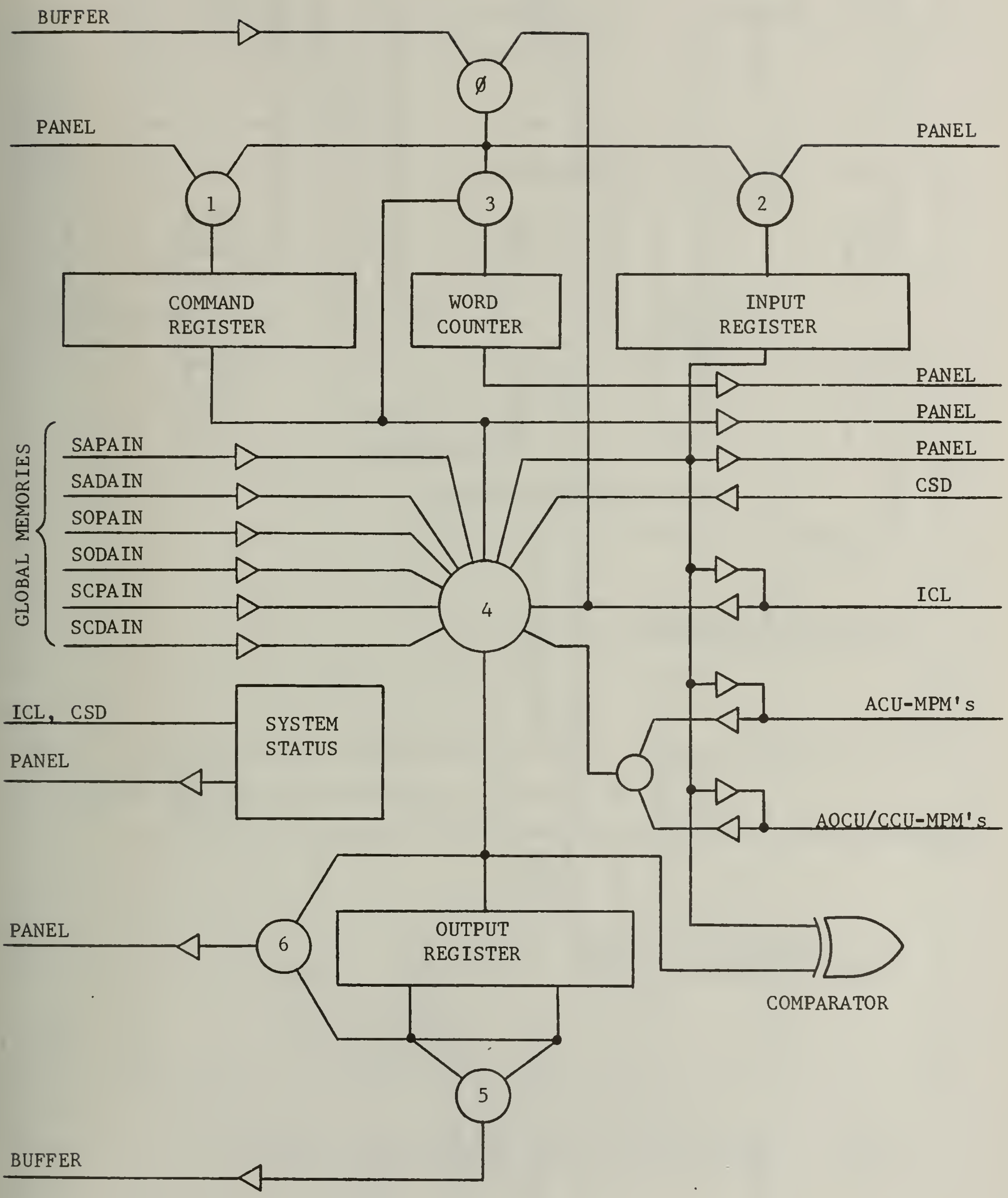

Figure 5. MCDU Block Diagram

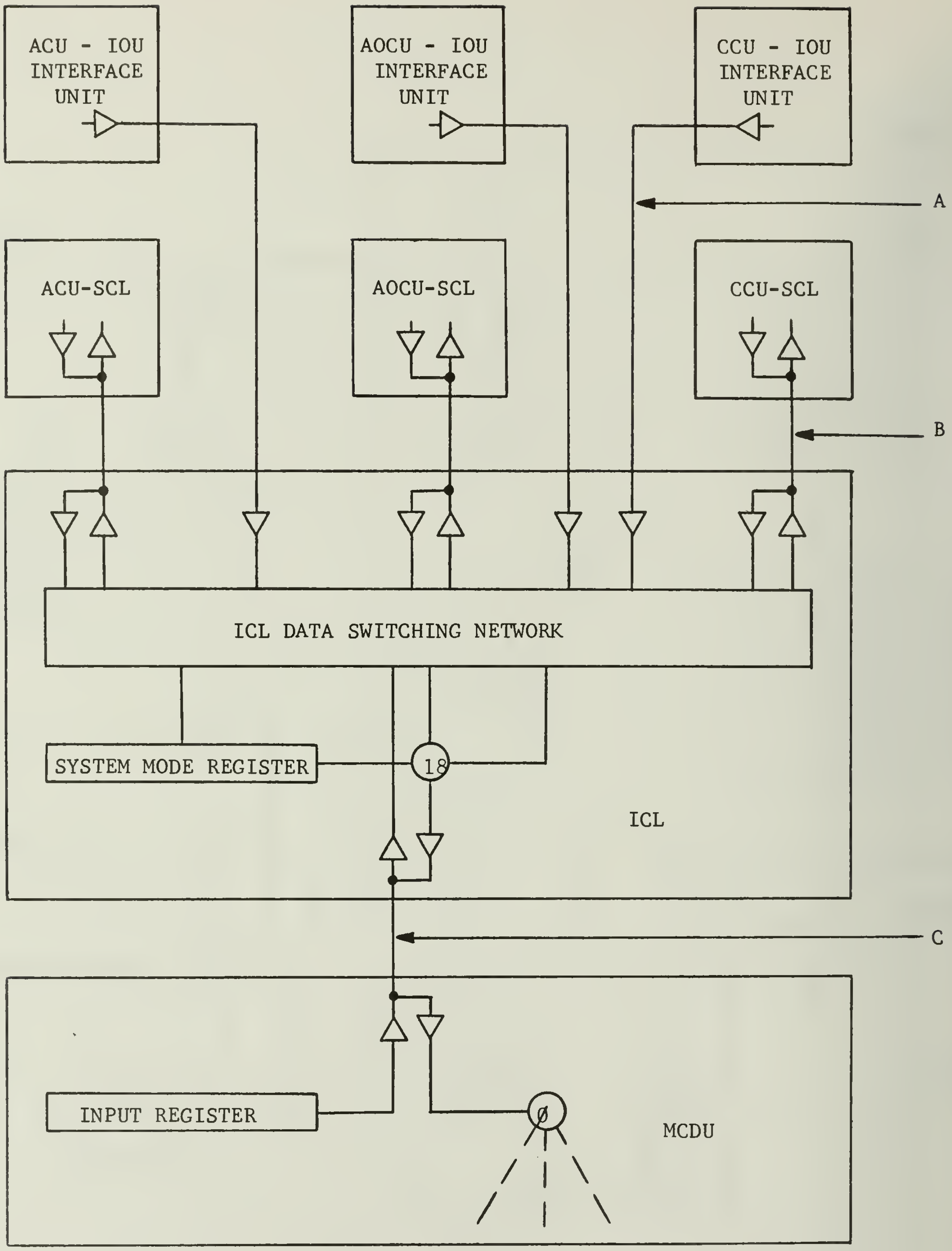

Figure 6. Major Control Console Busses Shared with MCDU

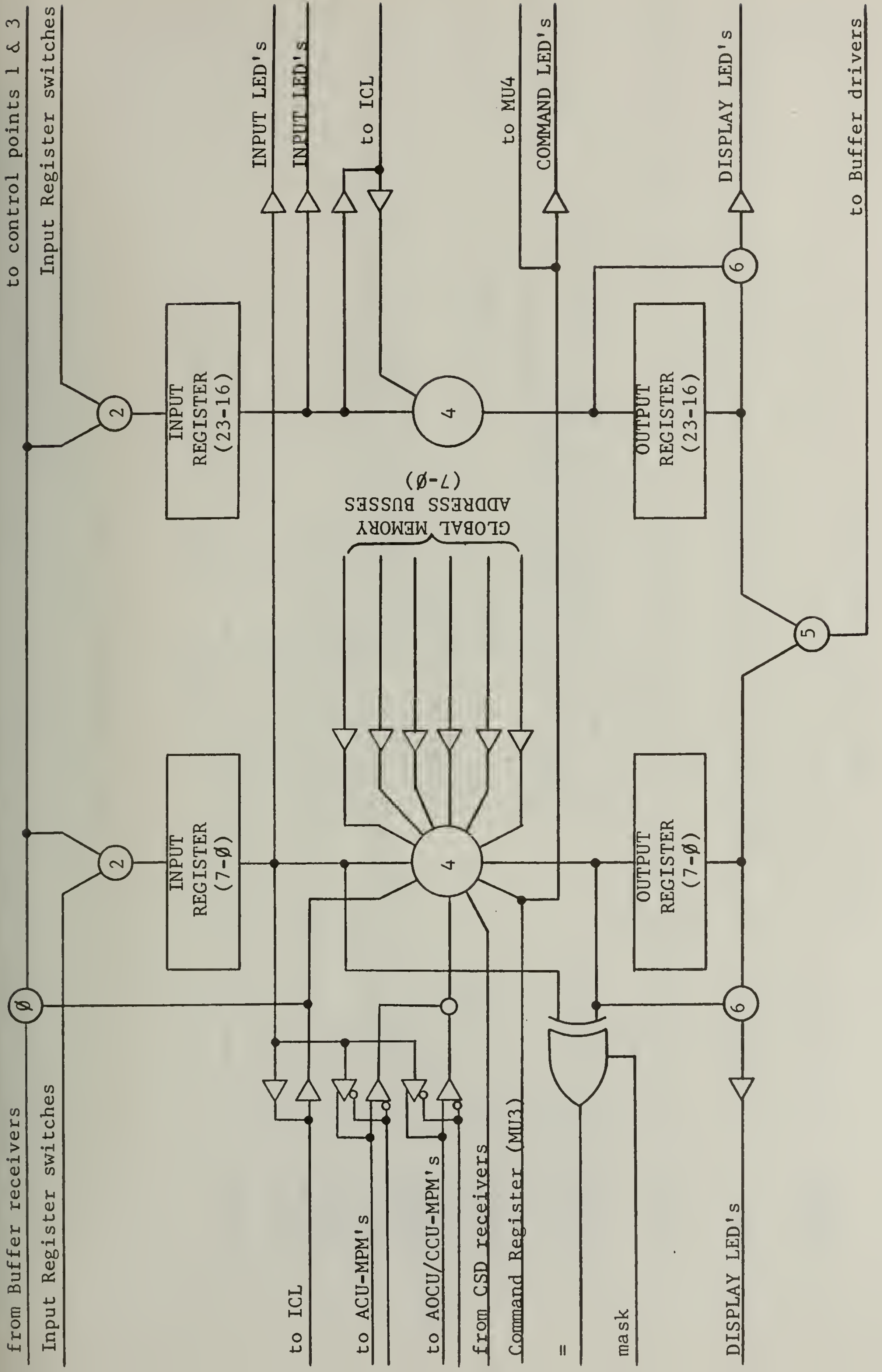

Figure 7. Data Board Logic Partition

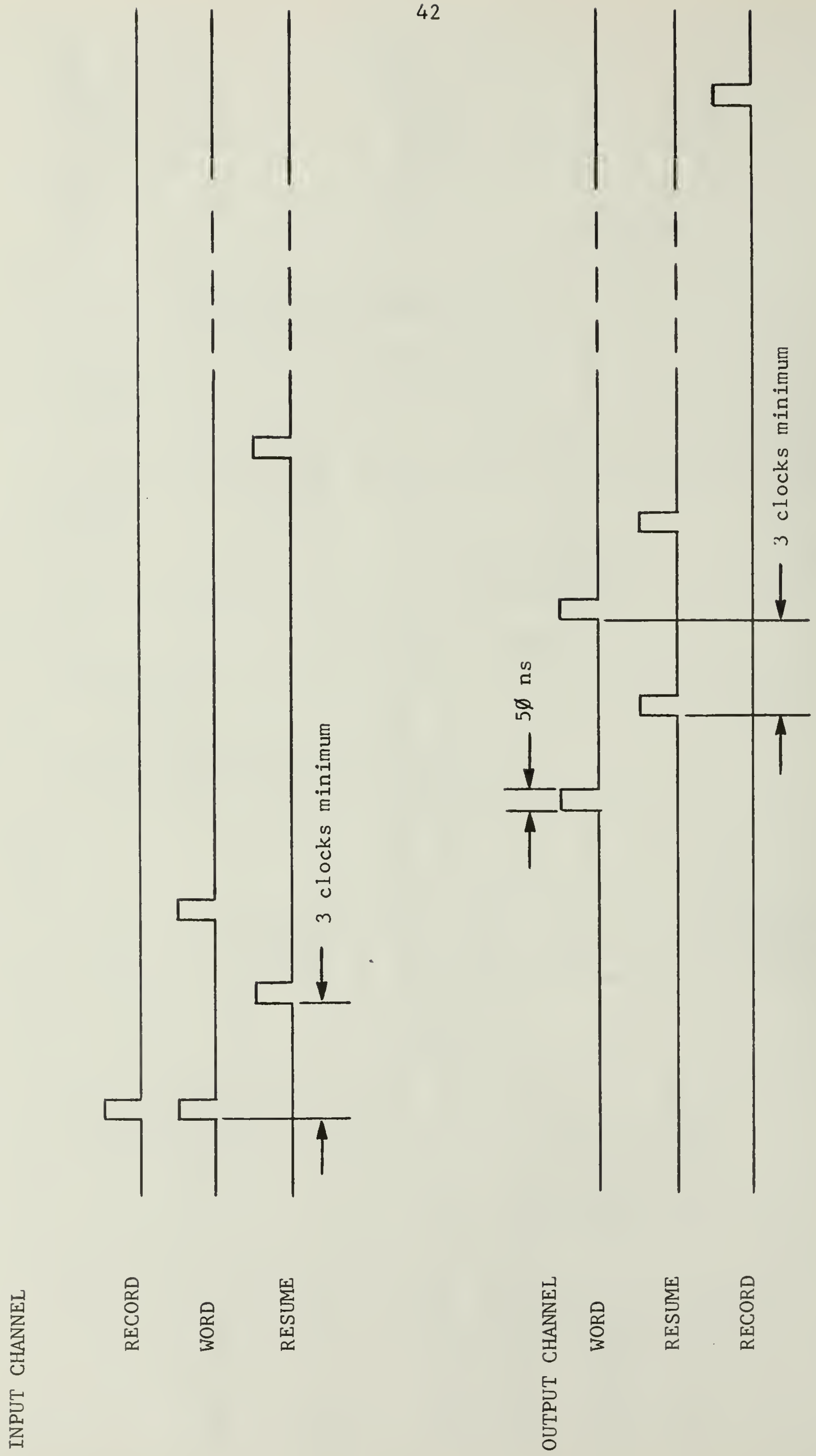

Figure 8. Typical Conversational Mode Timing

```
0000010 %CRQITRI
0000020 MSG CCU RQI & TRI, ACU RINTC INSTRUTION TEST
0000030 CLEAR ALL               %INITIALIZE PEPE STATE
0000040 XFER RQITRI ASTOP       %PAD ACU MEMORY TO STOP ON UNEXPECTED INTERRUPTS
0000050 XFER RQITRI ARINT MCDU  %LOAD CCU AND AOCU INTERRUPT HANDLERS IN ACU
0000060 XFER RQITRI CRQITRI     %LOAD CCU INTERRUPT TEST IN CCU
0000070 DO IBCD = M F *         %FIRST BLOCK WRITE TO MCDU
0000080 M 05 0 5 00             %START CCU AND ACU
0000090 (988) S 0000            %ZERO BUFFER MEMORY TO SIMPLIFY INSPECTION
0000100 END
0000110 WAIT 10                 %ALLOW 1 SECOND FOR I/O AND EXECUTION
0000120 NOP                     %FORCE READ OF BUFFER CONTENTS
0000130 CMP 0 S 114005          %TEST BUFFER DESCRIPTOR FOR GOOD COMPLETION
0000140 CLEAR ALL               %BE SURE SYSTEM NOT LEFT HUNG
```

Figure 9. Sample RAID Active File

```
   B1700 PEPE ASSEMBLER   LEVEL I.1   (01 JUL 75)   ASSEMBLY DATE = THURSDAY  01 APR 76

 INST WORD   ADDR  STMT                                       SOURCE  TEXT

00000000000 000000 0001  :$CONTROL
00000000000 000000 0002  :$PACKID PEPE2
00000000000 000000 0003  :$FILENAME RQITRI
00000000000 000000 0004  :          START ASTOP,ACU,0
25510000000 000000 0005  :          DC P,130*0@25510000000@
00000000000 000000 0006  :          END


TOTAL NUMBER OF SYNTAX ERRORS       0


00000000000 000030 0001  :          START ARINT,ACU,24
02004000000 000030 0002  :ORINT     SLD,S     SAXIR1,D
03104000002 000031 0003  :          STA,S     OINTTEST,D
00224000002 000032 0004  :          RINTO,P   OINTTEST,D
00410004000 000033 0005  :          SMD,S     0@4000@,I
25510000000 000034 0006  :          STOP
25510000000 000035 0007  :          DC P,3*0@25510000000@
02004000001 000040 0008  :CRINT     SLD,S     SAXIR2,D
03104000003 000041 0009  :          STA,S     CINTTEST,D
00124000003 000042 0010  :          RINTC,P   CINTTEST,D
00410004000 000043 0011  :          SMD,S     0@4000@,I
25510000000 000044 0012  :          STOP
00000400001 000000 0013  :SAXIR1    SCW D,2,1
00000400002 000001 0014  :SAXIR2    SCW D,2,2
00000000000 000002 0015  :OINTTEST DC D,0
00000000000 000003 0016  :CINTTEST DC D,0
00000000000 000000 0017  :          END


TOTAL NUMBER OF SYNTAX ERRORS       0
```

Figure 10. Sample SURPASS Output

```
   B1700 PEPE ASSEMBLER   LEVEL I.1   (01 JUL 75)   ASSEMBLY DATE = THURSDAY  01 APR 76

 INST WORD   ADDR   STMT                                  SOURCE  TEXT

00000000000 000000 0001  :          START CRQITRI,CCU,0
02410200000 000000 0002  :RQITEST   LDX,S,1    0,I
02410400000 000001 0003  :          LDX,S,2    0,I
02410600000 000002 0004  :          LDX,S,3    0,I
02110000001 000003 0005  :          LDA,S      1,I
03104000000 000004 0006  :LOOP1     STA,S      RESPONSE,D
00304000000 000005 0007  :          RQI,S      RESPONSE,D
02004000002 000006 0008  :          SLD,S      SAXIR6,D
37004000000 000007 0009  :          CMPL,S     RESPONSE,D
13110000012 000010 0010  :          BZ,S       NEXT1,I
11713400052 000011 0011  :          RTJX,S,14  ERROR,I
24010200001 000012 0012  :NEXT1     ADX,S,1    1,I
05410200040 000013 0013  :          CMPX,S,1   32,I
13110000020 000014 0014  :          BZ,S       TRITEST,I
02104000000 000015 0015  :          LDA,S      RESPONSE,D
17310177777 000016 0016  :          SHL,S      -1,I
11010000004 000017 0017  :          JMP,S      LOOP1,I
02410200000 000020 0018  :TRITEST   LDX,S,1    0,I
02110000001 000021 0019  :          LDA,S      1,I
03104000000 000022 0020  :LOOP2     STA,S      RESPONSE,D
00504000000 000023 0021  :          TRI,S      RESPONSE,D
02410400000 000024 0022  :          LDX,S,2    0,I
02010000000 000025 0023  :LOOP3     SLD,S      0,I
24204000003 000026 0024  :          ANA,S      =0@10000000@,D
00010000000 000003 0024  :    -- LITERAL --
37010000000 000027 0025  :          CMPL,S     0,I
13110000035 000030 0026  :          BZ,S       NEXT2,I
```

Figure 10. Sample SURPASS Output (continued)

```
24010400001 000031 0027  :           ADX,S,2    1,I
05410400012 000032 0028  :           CMPX,S,2   10,I
13610000025 000033 0029  :           BNZ,S      LOOP3,I
11713400052 000034 0030  :           RTJX,S,14  ERROR,I
02004000002 000035 0031  :NEXT2      SLD,S      SAXIR6,D
37004000000 000036 0032  :           CMPL,S     RESPONSE,D
13110000041 000037 0033  :           BZ,S       NEXT3,I
11713400052 000040 0034  :           RTJX,S,14  ERROR,I
24010200001 000041 0035  :NEXT3      ADX,S,1    1,I
05410200040 000042 0036  :           CMPX,S,1   32,I
13110000047 000043 0037  :           BZ,S       QUIT,I
02104000000 000044 0038  :           LDA,S      RESPONSE,D
17310177777 000045 0039  :           SHL,S      -1,I
11010000022 000046 0040  :           JMP,S      LOOP2,I
02110600000 000047 0041  :QUIT       TXA,S,3
03224000001 000050 0042  :           WRIU,S     AREG (1,0,1),D
25510000000 000051 0043  :           STOP
24010600001 000052 0044  :ERROR      ADX,S,3    1,I
03204000001 000053 0045  :           WRIU,S     AREG (0,0,1),D
02104000000 000054 0046  :           LDA,S      RESPONSE,D
03204000001 000055 0047  :           WRIU,S     AREG (0,0,1),D
02113400000 000056 0048  :           TXA,S,14
03204000001 000057 0049  :           WRIU,S     AREG (0,0,1),D
11013400000 000060 0050  :           JMP,S      0,I,14
00000000000 000000 0051  :RESPONSE   DC D,0
30005577777 000001 0052  :AREG       CCW D,3,1,3,D,-1
00000400006 000002 0053  :SAXIR6     SCW D,2,6
00000000000 000000 0054  :           END

TOTAL NUMBER OF SYNTAX ERRORS        0
```

Figure 10. Sample SURPASS Output (continued)

Table 1. RAID Process Verbs

```
         ⎧                                    ⎧ octal-word                    ⎫ ⎫
         ⎪ PEPE-mnemonic   [address]          ⎨                               ⎬ ⎪
LOAD     ⎨                                    ⎩ PEPE mnemonic instruction     ⎭ ⎬
         ⎪ MPM-mnemonic     address             octal-MPM-word                  ⎪
         ⎩                                                                      ⎭

         ⎧ PEPE mnemonic   [address] ⎫
DUMP     ⎨                           ⎬  [octal expected response]
         ⎩ MPM mnemonic     address  ⎭

START         control-unit-specifier      [DELAY]

         ⎡                        ⎧ SLD                  ⎫      ⎤
         ⎢                        ⎪ ERROR                ⎪      ⎥
STOP     ⎢ [ALL]    ON            ⎨ ⎧ PM ⎫               ⎬  IN  ⎥   control-unit-
         ⎢                        ⎪ ⎨ DM ⎬ =  address    ⎪      ⎥   specifier
         ⎣                        ⎩ ⎩ EM ⎭               ⎭      ⎦

CLEAR         [control-unit-specifier]

STEP     [MICRO]   control-unit-specifier      [step-count]

REPEAT        control-unit-specifier

SIOU          control-unit-specifier         IOU-control-function

NOP

LIT      [IBCD  =  buffer parameters]     octal-word-specifier

DO       [cycle count]   [IBCD  =  buffer parameters]

END
```

$$\underline{\text{MSG}}\quad \left[\text{message}\right]$$

$$\underline{\text{WAIT}}\quad \left[\text{interval in tenths of a second}\right]$$

$$\underline{\text{CMP}}\quad \text{start-buffer-address}\quad \text{octal-word-specifier}$$

$$\underline{\text{XFER}}\quad \text{file-name}\quad \text{segment-name}\quad \left[\underline{\text{MCDU}}\right]$$

where:

control-unit-specifier = $\underline{\text{A}}$, $\underline{\text{O}}$, $\underline{\text{C}}$, $\underline{\text{AO}}$, $\underline{\text{AC}}$, $\underline{\text{OC}}$, $\underline{\text{ALL}}$.

$$\text{octal-word-specifier} = \left[\left(\begin{array}{c}\text{replication}\\ \text{factor}\end{array}\right)\right] \left\{\begin{array}{ll}\underline{\text{M}} & \text{formatted MCDU command}\\ \underline{\text{S}} & \text{6 octal digits (16 bits)}\\ \underline{\text{D}} & \text{11 octal digits (32 bits)}\\ \underline{\text{E}} & \text{20 octal digits (60 bits)}\\ \underline{\text{I}} & \text{formatted IOU command}\end{array}\right\}$$

$$\text{buffer parameters} = \left\{\begin{array}{c}\underline{\text{M}}\\ \underline{\text{A}}\\ \underline{\text{O}}\\ \underline{\text{C}}\end{array}\right\} \left\{\begin{array}{c}\underline{\text{F}}\\ \underline{\text{M}}\\ \underline{\text{L}}\\ \underline{\text{S}}\\ \underline{\text{C}}\end{array}\right\} \left\{\begin{array}{c}\text{word count}\\ \\ \underline{*}\ \text{(RAID computes}\\ \text{the count)}\end{array}\right\}$$

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-76-822	2.	3. Recipient's Accession No.

4. Title and Subtitle The Role of the MCDU in the Parallel Element Processing Ensemble (PEPE)	5. Report Date August 1976
	6.
7. Author(s) Charles Paul Yonko	8. Performing Organization Rept. No. UIUCDCS-R-76-822
9. Performing Organization Name and Address Department of Computer Science University of Illinois at Urbana-Champaign Urbana, Illinois 61801	10. Project/Task/Work Unit No.
	11. Contract/Grant No.
12. Sponsoring Organization Name and Address Burroughs Corporation Paoli, PA 19301	13. Type of Report & Period Covered Master of Science Thesi
	14.

15. Supplementary Notes

16. Abstracts The Parallel Element Processing Ensemble (PEPE) consists of the Control Console and up to eight bays of Processing Elements, each element containing three computational units and a common memory. The Control Console is organized as three essentially independent control units and common logic available for inter-control unit communication. The Maintenance, Control and Diagnostic Unit (MCDU) is the part of the common logic which provides an interface between the Control Console and (a) the MCDU Panel, (b) the external Test and Maintenance (T&M) Computer via the PEPE Interface Buffer MCDU port, and (c) the Host or T&M computers via one of the normal PEPE Input/Output Units (IOU). These interfaces are established so that the MCDU can assist in performing initial system debug and integration, system hardware maintenance, failure diagnosis, firmware initialization, and real-time system status observation.

17. Key Words and Document Analysis. 17a. Descriptors

Digital Computers
- Programming Languages
  - Machine Languages
  - Programming Techniques
- Logic Design
- Programs (Computer)
  - Diagnostic Routines

17b. Identifiers/Open-Ended Terms

Parallel Element Processing Ensemble
PEPE
Parallel Processing
Maintenance, Control, and Diagnostic Unit

17c. COSATI Field/Group

18. Availability Statement Release Unlimited	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 51
	20. Security Class (This Page) UNCLASSIFIED	22. Price ------